National University of Computer and Emerging Sciences, Islamabad

Spring 2024

Parallel and Distributed Computing

Assignment No. 3

Submitted By:

Name: Syeda Areeba Nadeem
Section: AI-K
Roll No: 21I-0307

Submitted To:

Dr. Qaiser Shafi

1. <u>Host Program Description:</u>

- The host program uses OpenCL to convert RGB images to grayscale.
- It starts by initializing OpenCL, creating a context, command queue, and memory buffers for input and output images.
- The grayscale conversion algorithm is implemented in an OpenCL kernel called "RGB_to_Gray".
- The program reads input images from a specified folder ("ISIC_2020_Test_Input") and saves the grayscale images to another folder ("ISIC_2020_Test_Output").
- The conversion process is performed for each image in the input folder, and the resulting grayscale images are saved in the output folder.
- Memory cleanup is done after processing each image to release OpenCL resources and free allocated memory.

The host program is designed to convert an RGB image to grayscale using OpenCL. Here's a breakdown of the steps and strategies used:

Input and Output Handling:

- Input images are loaded using the STB Image library (stbi_load) from the input folder.
- Grayscale output images are saved using stbi_write_png in the output folder.
- Error handling is implemented to manage cases such as failed image loading or writing.

Configuration Steps:

- Image dimensions (width and height) are initialized with example values.
- Memory is allocated for the image data (RGB format) based on the dimensions.
- The program reads the input image data and stores it in memory.

OpenCL Initialization:

- Platform and device are selected using clGetPlatformIDs and clGetDeviceIDs.
- A context is created with the selected device.
- A command queue is established for kernel execution.

OpenCL Kernel and Program Creation:

- The grayscale conversion kernel is defined inline within the program.
- The program is created using clCreateProgramWithSource and built using clBuildProgram.
- The kernel function RGB_to_Gray is created with clCreateKernel.

Memory Buffers and Kernel Arguments:

- Input and output memory buffers are created using clCreateBuffer.
- Kernel arguments are set using clSetKernelArg to pass input and output buffers, as well as image dimensions to the kernel.

Kernel Execution:

- The kernel is enqueued for execution using clEnqueueNDRangeKernel, specifying the global work size based on image dimensions.

Result Handling:

- The output grayscale image data is read back from the device using clEnqueueReadBuffer.
- The resulting grayscale image is saved to a jpg image.

Clean-up:

- Memory allocated for image data and output data is freed.
- OpenCL resources (buffers, kernel, program, command queue, context) are released to avoid memory leaks.

2. Grayscale Conversion Algorithm:

The RGB to grayscale conversion algorithm used in the OpenCL kernel is a standard luminosity method:

$$Gray = 0.299 \times Red + 0.587 \times Green + 0.114 \times Blue$$

Algorithm Explanation:

- The kernel accesses each pixel's RGB values using global IDs and indices.
- The grayscale value is computed using the specified weights, according to formula, for each color channel (R, G, B).

3. OpenCL Kernel Design:

The OpenCL kernel rgb_to_gray is designed to operate on individual pixels of the input RGB image and convert them to grayscale using the luminosity method. Key aspects of the kernel design include:

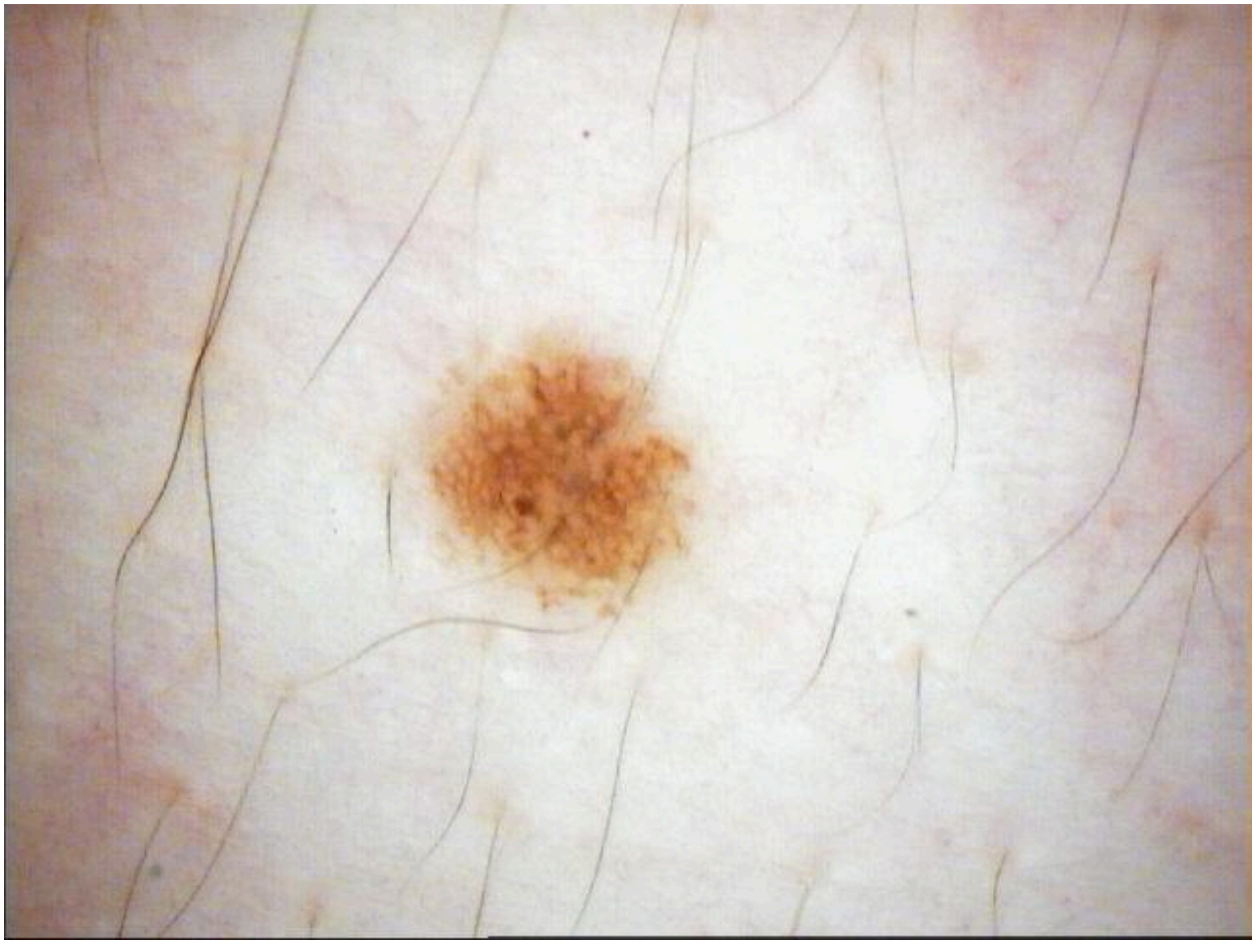
Input Parameters:

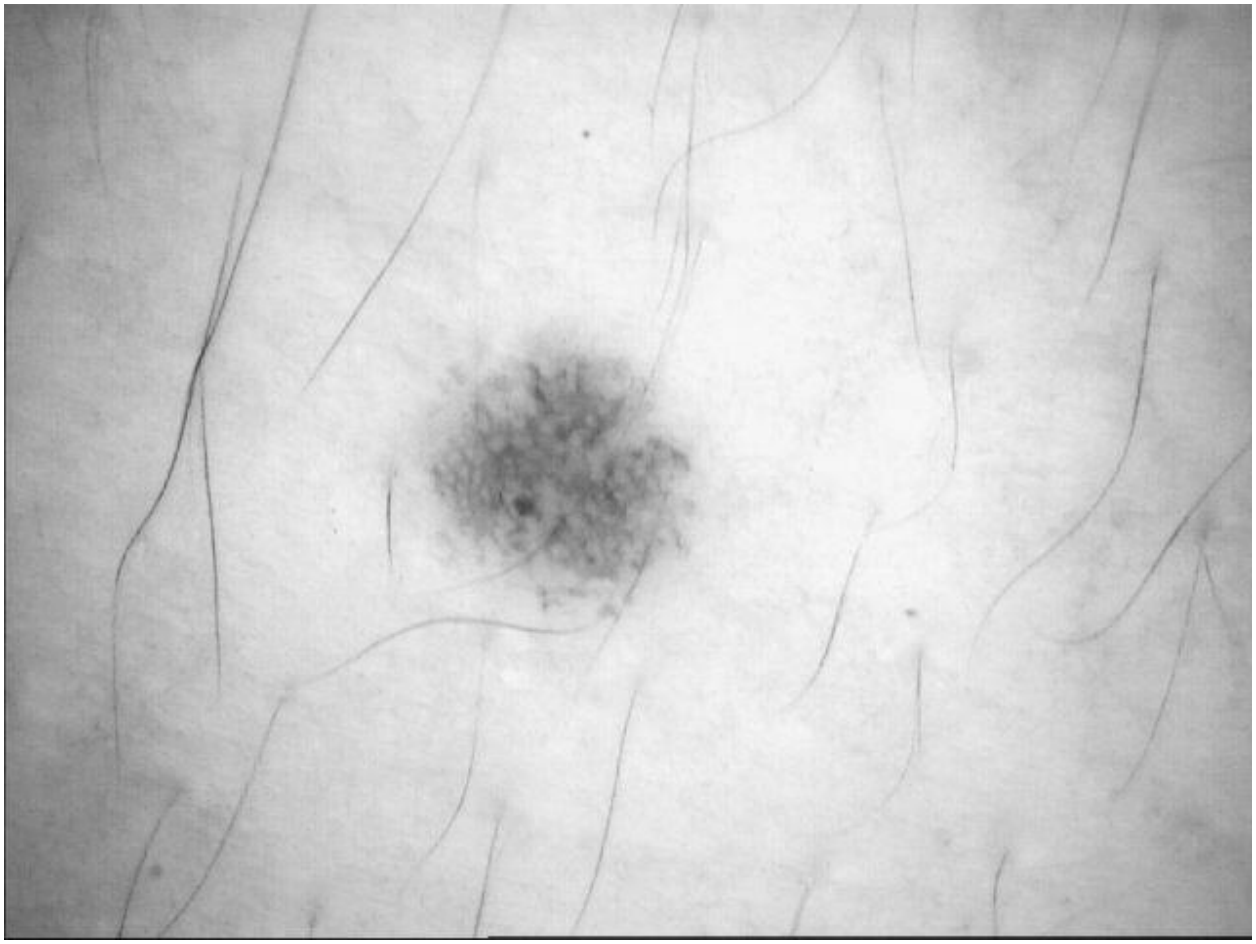
- __global unsigned char *input: Pointer to the input RGB image data.
- __global unsigned char *output: Pointer to store the output grayscale image data.
- int width, int height, int channels: Dimensions & channels (RGB) of the image.

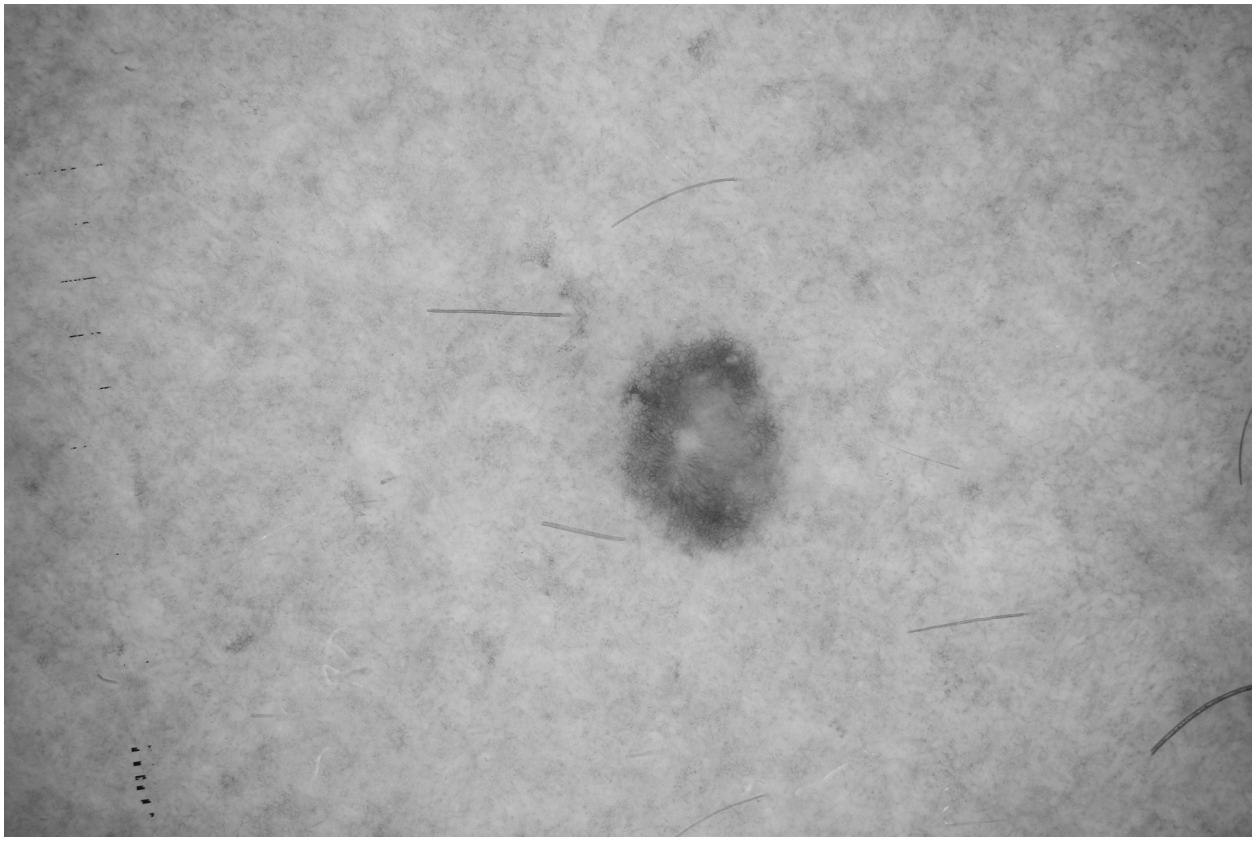
Pixel Processing:

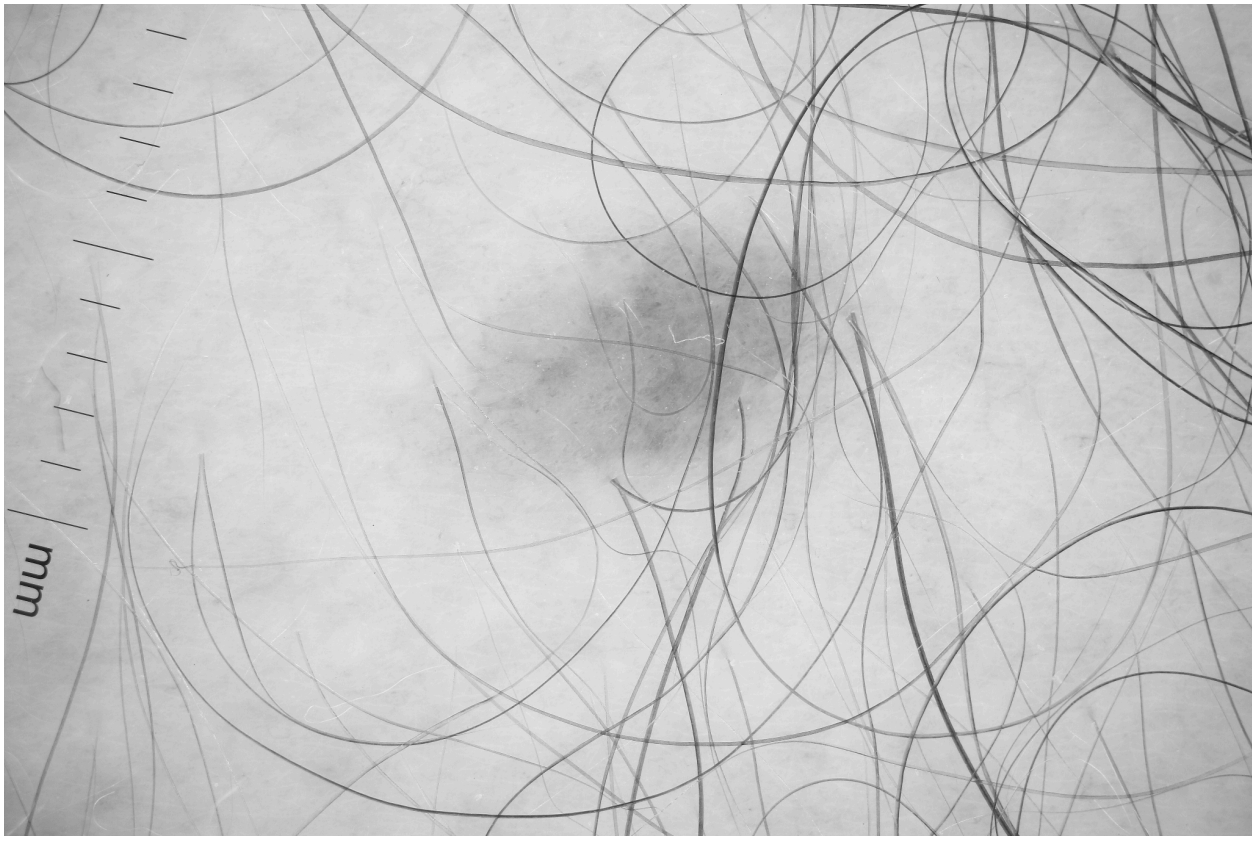
- Global IDs are used to determine the current pixel's position in the image.
- RGB values are extracted from the input buffer based on the pixel's index.
- Grayscale value is computed using the luminosity formula and stored in the output buffer.

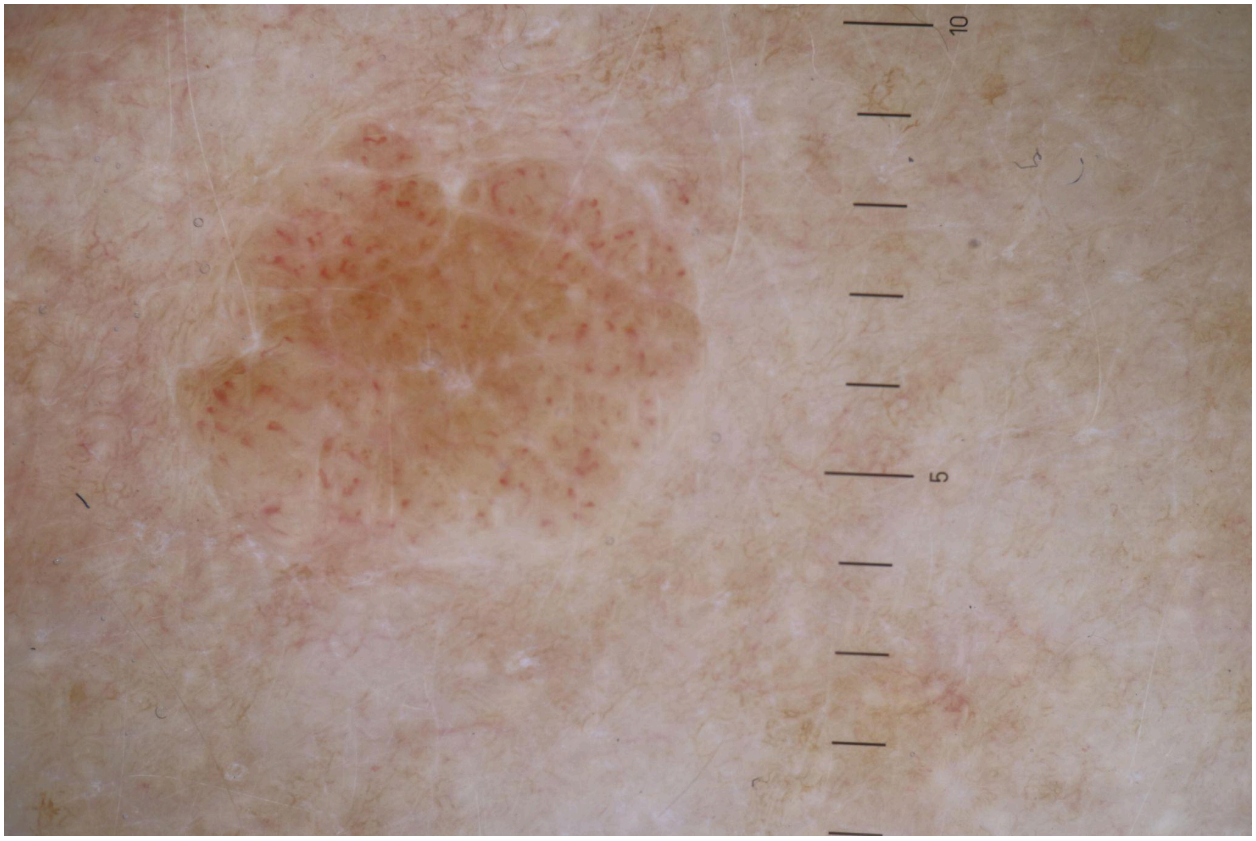
4. Results:

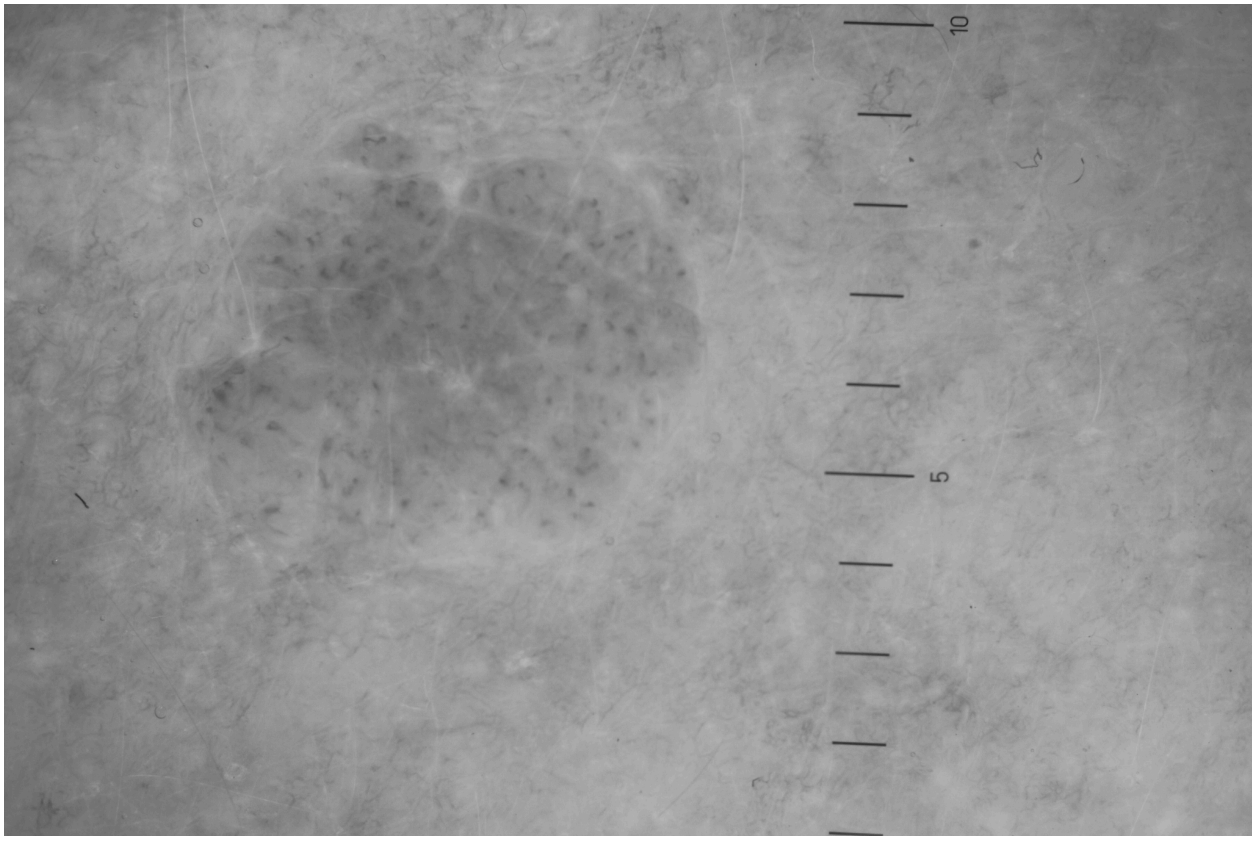


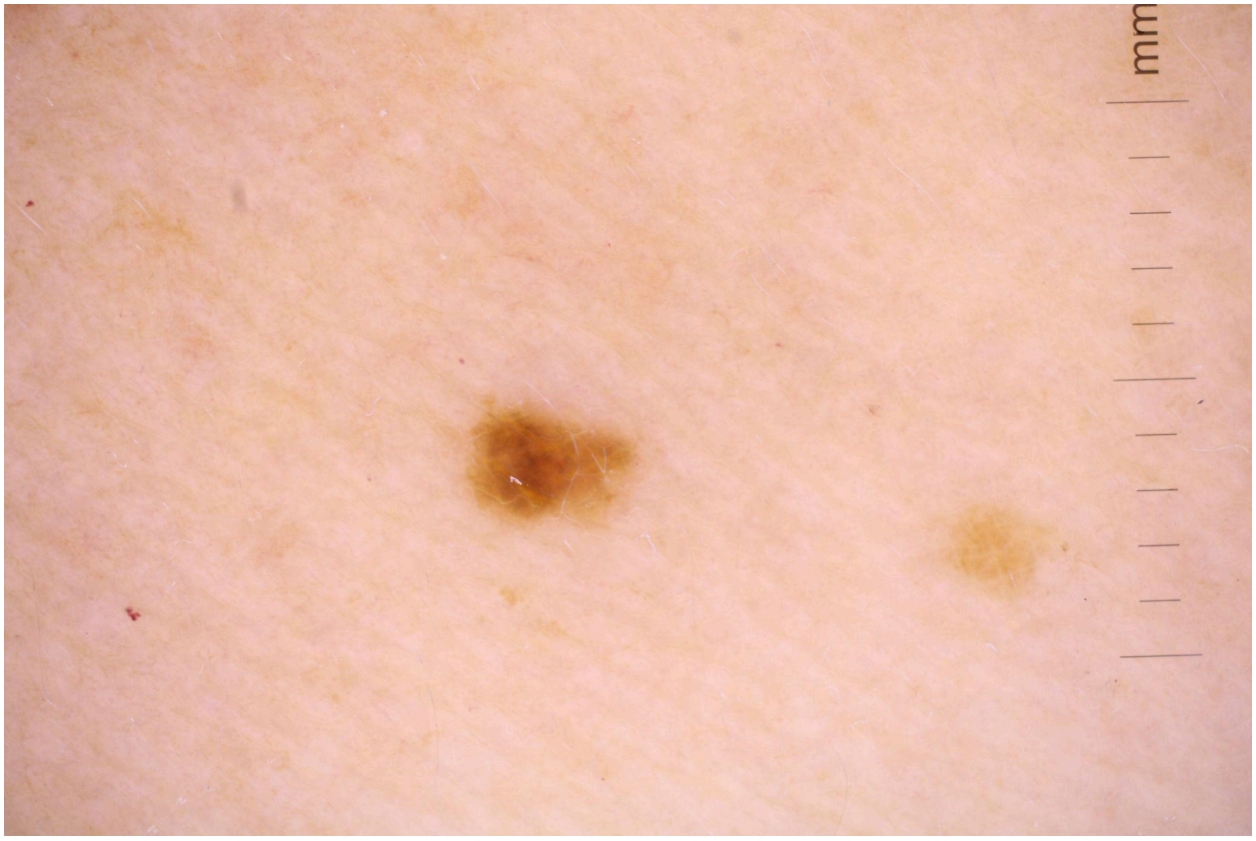


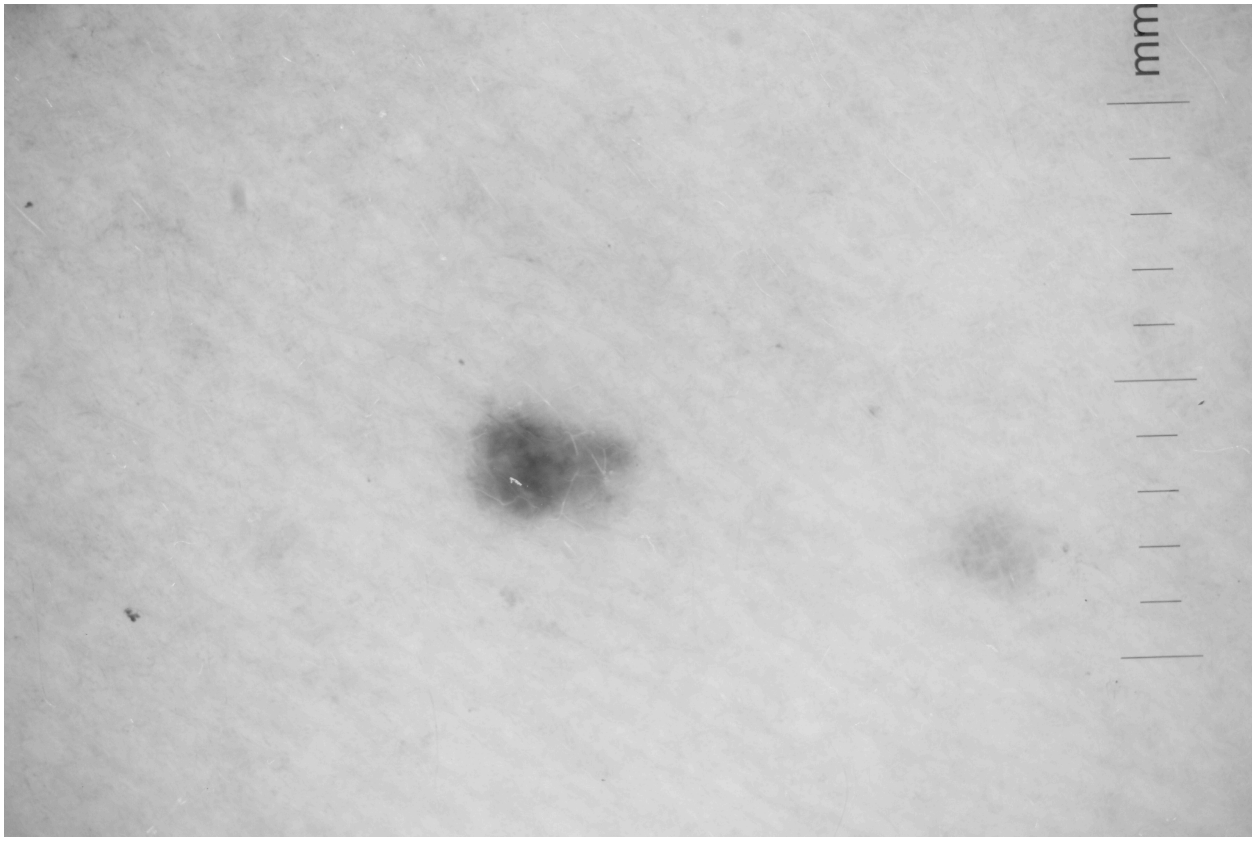





10
5


10
5


mm


mm